DONNAv2 - Lightweight Neural Architecture Search for Vision tasks

Sweta Priyadarshi, Tianyu Jiang, Hsin-Pai Cheng, Sendil Krishna, Viswanath Ganapathy, Chirag Patel
Qualcomm AI Research*
San Diego, CA, USA 92121
{swetpriy, tianyuj, hsinpaic, sendilk, viswgana, cpatel}@qti.qualcomm.com

Abstract

With the growing demand for vision applications and deployment across edge devices, the development of hardware-friendly architectures that maintain performance during device deployment becomes crucial. Neural architecture search (NAS) techniques explore various approaches to discover efficient architectures for diverse learning tasks in a computationally efficient manner. In this paper, we present the next-generation neural architecture design for computationally efficient neural architecture distillation - DONNAv2 . Conventional NAS algorithms rely on a computationally extensive stage where an accuracy predictor is learned to estimate model performance within search space. This building of accuracy predictors helps them predict the performance of models that are not being finetuned. Here, we have developed an elegant approach to eliminate building the accuracy predictor and extend DONNA to a computationally efficient setting. The loss metric of individual blocks forming the network serves as the surrogate performance measure for the sampled models in the NAS search stage. To validate the performance of DONNAv2 we have performed extensive experiments involving a range of diverse vision tasks including classification, object detection, image denoising, super-resolution, and panoptic perception network (YOLOP). The hardware-in-the-loop experiments were carried out using the Samsung Galaxy S10 mobile platform. Notably, DONNAv2 reduces the computational cost of DONNA by 10x for the larger datasets. Furthermore, to improve the quality of NAS search space, DONNAv2 leverages a block knowledge distillation filter to remove blocks with high inference costs.

1. Introduction

Computer vision algorithms are being widely deployed on edge devices for several real-world applications including medicine, XR-VR technology, visual perception, and autonomous driving. However, computer vision algorithms based on deep learning require significant computational resources. Therefore, efficient search for deep learning architecture has attracted a lot of attention. Most of these NAS efforts are agnostic to the requirements of resource-constrained edge devices. Further, current NAS methods that operate over large search spaces are computationally very expensive to generate the optimized models. NAS based on block knowledge distillation (BKD) [2, 20, 10] scales well over large search spaces in a computationally efficient manner. In this work, we leverage BKD for hardware-aware NAS. The core process of our NAS approach based on BKD consists of building replacement blocks, building accuracy predictors, predicting the accuracy of models, and based on their cost(flops, parameters, latency on hardware), the models are picked based on the trade-off between predicted accuracy and cost. In multiple studies, it appears that the stage of building the accuracy predictor is the most expensive bottleneck of the pipeline. Researchers have worked on making the accuracy predictor stage efficient by utilizing regression or ranking methods. Nevertheless, the majority of the computation time for the NAS pipeline is still taken up by the accuracy predictor stage. Our work DONNAv2 aims at reducing the search space by identifying the redundant blocks and eliminating them from the search space. We have defined this method as the Blockwise Knowledge Distillation filtering stage. Furthermore, we aimed at removing the accuracy predictor stage, which was by far the most computationally expensive component of the DONNA pipeline. Our work DONNAv2 brings a more sophisticated method of approximating blockwise losses to network losses.

Many NAS studies have focused on optimizing models for hardware-agnostic complexity metrics like flops (MACs). But some of the analyses [9], indicate that flops do not always translate linearly to the latency or the power of the model. To find the best architecture for a given use case, and a given hardware, it is important to specifically optimize models to minimize latency and energy consumption for on-device performance. Many NAS performs with a lookup ta-

[*] Qualcomm AI Research is an initiative of Qualcomm Technologies, Inc.

ble that is reporting per-layer latency and is approximated to full model latency. Here, the assumption is that the linear sum of latency would be model latency which does not hold true always. We have hardware in the loop to optimize models for a given hardware. But unlike many expensive methods, DONNAv2 tends to provide optimal neural networks at a lower complexity for a similar diverse search space. In our work, we have compared the time complexity saved by pivoting to the approximation method using mean square error (MSE) loss rather than training an accuracy predictor model.
We have described our paper through the DONNAv2 pipeline that comprises of - Block knowledge Distillation (BKD), BKD Filtering, Evolutionary Search, and Finetuning for a Galaxy S10 mobile platform. Finally, we extend our paper to cover five vision tasks to show how DONNAv2 led to an optimal compressed model without losing accuracy. The vision tasks highlighting the benefits of DONNAv2 are but not limited to image classification, object detection, super-resolution, image denoising, and multitask network.

2. Related Work

We can delve into the historical progression of NAS to trace its evolution from initially computationally expensive methods involving diverse search space [24, 31, 32] to low computation methods with very small search space [3, 23]. DONNA [20], explored approaches to reduce the computational burden using a block-based search space. Recent study [5] has also validated the efficacy of NAS approach developed in DONNA. Here, DONNAv2 , we aim to further reduce the computation time of the search while keeping the search space similar to DONNA. Mobile neural architecture search (MNAS) [24] is an expensive method that requires around 40,000 epochs to perform a single search. Other attempts for NAS included differentiable architecture search methods such as DARTS [17], FBNet [28], FBNetV2 [27], ProxylessNAS [4], AtomNAS [18] and Single-Path NAS[23] that simultaneously optimize the weights of a large super-net and its architectural parameters. However, in these cases the granularity of the search level suffers and methods need to be repeated in every scenario or when the search space changes. There have been studies [1, 11, 19, 10] to construct proxies for ranking models using the search space. These include attempts based on zero-shot proxy [1] and one-shot proxy NAS [11]. A similar approach LANA [19], also leverages the loss function as a proxy method to rank the model. A recent work [10] explores a hardware-aware search by translating the multi-objective optimization problem into a combinatorial optimization problem. However, this approach assumes chain-structured NAS and is not readily applicable to more general architectures. Our DONNAv2 builds on the idea of using the loss function as the proxy with the following enhancements:

- enables hardware-aware search in a diverse search space with the hardware in the loop for latency measurements. Earlier studies leveraged a linear sum of the pre-computed feature layer latencies to estimate the latency of a deep learning model. However, this does not capture the true latency when compilers leveraged a depth-first search.
- DONNAv2 is scalable when the search is expanded or the hardware platform changes.
- DONNAv2 converges 9x faster during the finetuning stage while achieving a similar accuracy compared to training-from-scratch ([14]).

3. DONNAv2 - Lightweight NAS

DONNAv2 follows the steps in DONNA, while eliminating the accuracy predictor stage and introducing a blockwise knowledge distillation (BKD) filtering stage. In DONNAv2 , we start by defining a search space and then building a BKD library which gets further filtered out by a BKD filter. These filtered blocks from the BKD library are utilized by the evolutionary search phase to find the Pareto-optimal network architectures for any specific scenario using the loss metric of individual blocks. Finally, the predicted Pareto-optimal architectures are fine-tuned to full accuracy for deployment.

3.1. Search Space

Search Space in DONNAv2 follows a block-level architecture and only parameters within the blocks are varied. A collection of blocks to build candidate networks are generated based on user-defined blocks and the associated parameters. To determine a suitable search space, we include diverse macro-architectural network parameters such as layer types, attention mechanisms, and channel widths. Furthermore, micro-architectural parameters such as cell repeats within a block, kernel sizes of layers within cells, and in-cell expansion rates were also utilized. In our experiments, the cardinality of the search space was of the order of 1e14. The larger the search space, the higher the chances of identifying hardware-friendly and performance-achieving networks.

3.2. Blockwise Knowledge Distillation

Blockwise Knowledge Distillation (BKD) is the first building block of the lightweight NAS, DONNAv2 . Unlike DONNA, DONNAv2 uses BKD not for building an accuracy predictor, but as an input to produce a surrogate metric to generate the Pareto optimal curve. The BKD stage generates a Block Library with pre-trained weights and loss metrics for each of the option blocks $B_{n,m}$ that is used as

[h]

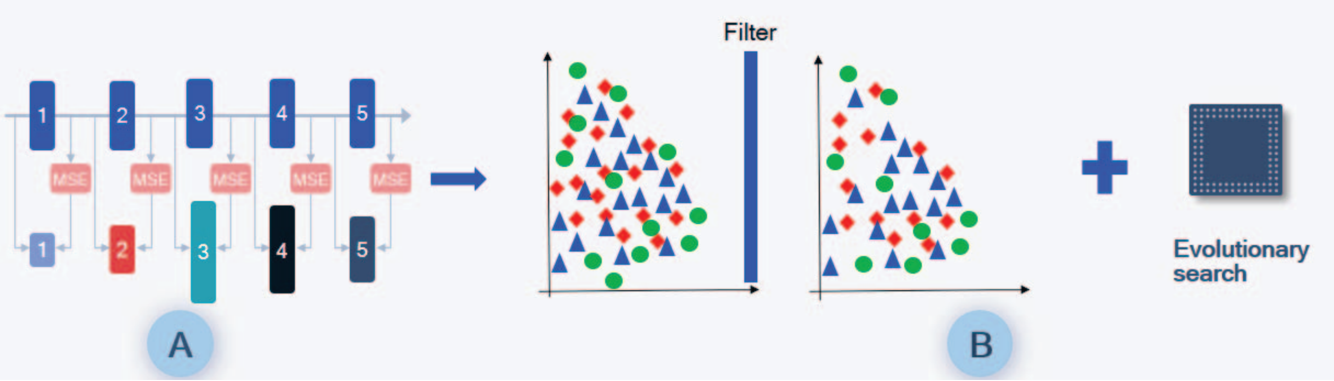


Figure 1: DONNAv2 Pipeline - includes Stage A which is composed of search space definition and Block Knowledge Distillation(BKD), and Stage B which includes BKD Filtering, Evolutionary search(hardware in the loop), and Finetuning.

the replacement. To build the BKD library, each block option $B_{n,m}$ is replaced in the blocks of the mothernet and trained as a student model using the mothernet block B_n as a teacher. The MSE loss between the teacher's output feature map Y_n and the student's output feature map $\bar{Y}_{n,m}$ is used as the surrogate metric in the evolutionary search stage and the BKD filtering stage. One epoch of complete dataset training is employed at this stage for building the BKD library and is denoted as 1e. The pre-trained weights at this stage help in faster convergence while finetuning the model.

3.3. Blockwise Knowledge Distillation Filtering

Blockwise Knowledge Distillation Filtering method aims to identify and drop the inefficient blocks based on the optimization strategy. Here, the optimization strategy is defined as the cost of the optimized model in terms of flops, latency, power consumption, etc. The BKD filtering stage retains only blocks with a minimum cost ratio with respect to the associated blocks in the reference model. Blocks with minimum cost ratio will retain performance-achieving efficient candidate models during the evolutionary search. The cost ratio of a block is estimated for a given loss metric. Retaining only blocks with the best cost ratio reduces the number of blocks and thereby the cardinality of the search space. However, it is important to note that block filtering does not eliminate good models in the sample space. We have validated this with experiments across several learning tasks. In Figure 3, the legend id tells the blocks of a particular layer we are filtering and the blue dots represent the blocks that are retained and grey dots represent the blocks that would be dropped.

In Algorithm 1, we have described the steps in detail.

Algorithm 1 BKD filtering

```
Input: BKD library, threshold = D.
       $B_{(n,m)}$ is the $m^{th}$ potential replacement out of M
       choices for block $B_n$ in the mothernet model.
for i do = 1 to m do
    $L_{(n,m)}$ = Calculate the block $B_{(n,m)}$ inference cost of model
    $MSE_{(n,m)}$ = Calculate the block $B_{(n,m)}$ MSE loss of model w.r.t mothernet
    $C_{(n,m)}$ = Calculate the ratio of $L_{(n,m)}$ w.r.t mothernet block inference cost
    Plot cost ratio vs MSE on a plot
    Discard the blocks at each MSE loss with higher inference cost based on the threshold D.
    OUTPUT: Obtain new BKD filtered library
end for
```

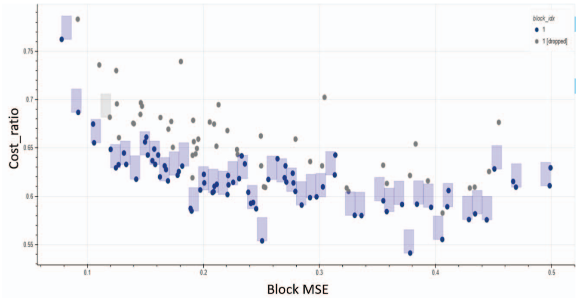
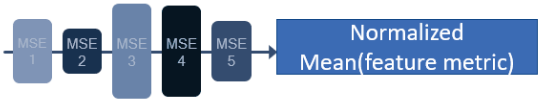


Figure 2: BKD Filtering - The x-axis is the surrogate loss metric and the y-axis is the cost ratio(flops/latency on device) between the replacement blocks and the mothernet

3.4. Evolutionary Search

Evolutionary search utilizes the MSE loss metric as a surrogate measure. Here, in contrast to DONNA, we lack



[h]

Figure 3: Metric developed using the MSE loss of the blocks forming the network, for the Pareto optimal curve

the predicted accuracy of the candidate models from the Pareto front. The performance of the model is approximated as the sum of the MSE of the blocks constituting the model. We built the Pareto optimal curve by using this surrogate measure of the model. Given the MSE loss metric of blocks from the block library and latency of the models formed by using the block options, the NSGA-II evolutionary algorithm is leveraged to find Pareto-optimal architectures that minimize the model loss and cost. The cost utilized could be scenario-agnostic measures such as the number of operations (MAC) or the number of parameters in the network (params). The scenario-aware cost includes on-device latency, cycles of operation, and energy. In our experiments, we have utilized on-device latency as a cost function by using direct hardware measurements in the optimization loop. After obtaining the Pareto-optimal models, we selected the model with appropriate latency and finetuned the candidate model to obtain the final model.

3.5. Finetuning

Empirically it has been observed that the final architectures from the Pareto front curve converge faster than training from scratch when pre-trained with weights obtained from the BKD stage. It has been shown that EfficientNet-style models can converge in around 50 epochs as opposed to 450 epochs when trained from scratch.

4. Experiments & Results

In this section we will discuss the detailed experimental evaluation of DONNAv2 : across a set of diverse computer vision tasks. The performance of DONNAv2 for Image classification, Object detection and super-resolution tasks were quantified with the Samsung Mobile hardware platform in the loop. The performance of DONNAv2 for image denoising and multitask network was validated using the number of operations(MAC). Importantly, all experiments demonstrated significant model compression with minimal performance degradation on an edge device. The performance of DONNAv2 is captured in terms of accuracy, on-device latency/MAC, and the number of epochs (defined as the sum of the number of epochs for training accuracy predictor, the number of epochs used for finetuning and building the block library (1e)). It is important to note that there is very few NAS methodology that has worked for diverse vision tasks. Many research focuses on NAS method for individual Vision tasks, but we aim to focus on key components that remain the same across the wide range of vision tasks and deliver the same method to be applied across various vision tasks. Details of each of the experiments are described below:

4.1. Search Algorithm

In this section, we have summarized the overall DONNAv2 setup in an algorithm format as shown in algorithm 2. It provides step by step setup details to perform the BKD based searching.

Algorithm 2 DONNAv2 search

```
Input: Begin with a baseline or mothernet network.
    Split the baseline into stem, head and N blocks.
    $B_(n, m)$ is the $m^{th}$ potential replacement out of M
    choices for block $B_n$ in the baseline model.
  for i do = 1 to m do
      BKD:Replace block $B_n$ with $B_(n, i)$ and train the
          new architecture for 1 epoch of complete dataset.
          Complete the step for all blocks and all options
          and construct a BKD library with MSE loss of
          replacement blocks w.r.t the mothernet.
  end for
  BKD Filetring : Perform BKD filtering to remove redundant blocks as explained in Section 3.2
  Evolutionary Search:
  Input: Population Size= E, number of search steps = T
  BKD Library
  for i do = 1 to T do
      Randomly sample E networks Ft from networks composed of $B_(n, m)$ blocks
      Compute inference cost & MSE of the sampled model Ft
      Retain models with lowest MSE loss in each iteration at different computation cost.
  end for
  OUTPUT: Pareto optimal curve of models at different latency
  Pick model X and finetune.
```

4.2. Image Classification

We present experiments for DONNA search spaces for ImageNet [8] classification that was earlier discussed in DONNA [20]. The mothernet chosen here, was Efficientnet-B0 [25] style architecture with 6 blocks instead of 7. We searched over 5 blocks of the mothernet numbered 1 to 6 using DONNA search space. DONNA search space had a choice out of M=192 options: kernel size $k \in 3, 5$; expansion ratio $expand \in 2, 3, 4$; $depth \in 1, 2, 3, 4$; $activation \in ReLU/Swish$; $layer -$

$type \in grouped, depthwiseinvertedresidualbottleneck$; and $channel - scaling \in 0.5, 1.0$. The search space can be expanded or arbitrarily constrained to known efficient architectures for a device. Each of these $5 * 192 = 980$ alternative blocks is trained using BKD to complete the Block Library. At this end, we perform the BKD filtering to obtain 768 blocks, thus removing the remaining redundant 212 blocks. After preparing the filtered BKD library, we perform the NSGA-2 [7] algorithm-based search with 100 population size and 50 steps to obtain the Pareto optimal curve. We first show that networks found by DONNAv2 in the DONNA search space [20] outperform the network found by DONNA at similar latency[1]. DONNAv2 achieves similar accuracy at 10X less computational time. The table3 shows that the number of epochs for DONNAv2 is significantly lower than DONNA since there is no computation expended for training accuracy predictor. DONNAv2 reduces inference latency as well as model search cost. The model search cost reduction is significant for DONNAv2 since 2500 epochs on ImageNet would cost several GPU hours. Further, in Figure 5, we can see that DONNAv2 can identify efficient architectures across a similar latency range as DONNA. Table1 captures the comparison of our methodology against the popular NAS methods and it can be observed that our methodology DONNAv2 has lowered the computation cost drastically compared to other methods making it more usable by the research community to find more hardware friendly efficient models. The latency numbers reported in the table 3 are conducted on Samsung Galaxy S10 mobile platform. Figure 4, describes the efficacy of the block filtering step and compares models in the Pareto front for DONNA and DONNA v2. The left y-axis is the accuracy predictor stage and the right y-axis is the loss surrogate metric. The figure shows that DONNA v2 search, similar to DONNA, identifies wide range DNN models across the satisfying varying accuracy latency trade-offs. The diversity of models as shown in Figure 4 is similar for DONNAv2 and DONNA.

4.2.1 Performance analysis for classification task

Here, we attempt to leverage centralized kernel alignment (CKA), [13], to visualize the DONNAv2 optimized models. Further, we relate interaction between layers of DONNAv2 optimized models using CKA and the surrogate loss. The feature map similarities of CNNs have a block structure. Layers in the same block group (i.e. at the same feature map scale) are more similar than layers in different block groups. DONNAv2 surrogate loss leverages the loss metric of individual blocks forming the network as the performance predictor for the sampled models. CKA analysis of the models from the Pareto optimal curve for the image classification task is shown in figure (4). In Figure 4, we can observe that the heatmap of layers of the mothernet shows a checkerboard pattern displaying the local block level similarity. The similarity measure for the mothernet is confined to the local blocks. However, as we start pruning the layers using DONNAv2 , we can observe that for the model (d), a large big yellow box demonstrates similarity in representation across several layers. The fine-tuned accuracy of the model (d) also indicates the performance is saturated. Further, the fine-tuned accuracy of the DONNAv2 optimized models shown in the table 2 correlates with DONNAv2 surrogate loss. The CKA similarity shown in figure(4) also correlates with DONNAv2 surrogate loss.

4.3. Object Detection

Object detection is one of the dense vision tasks, on which extensive neural architecture search is performed. Here, we have identified NAS optimized model EfficientDet-D0 [26] as the baseline model to further optimize this model in terms of latency and accuracy. The search space identified here has been inspired by the image classification task, as we profiled the object detection model and identified that the majority of the latency of the model resides in the backbone contributing almost 60% of the end-to-end model. The backbone of the EfficientDet-D0 model is EfficientNet-B0. Hence, our search space for the object detection task includes kernel sizes $k \in 3, 5$; $expand \in 2, 3, 4, 6$; $depth \in 1, 2, 3, 4$; $layer - type \in grouped, depthwiseinvertedresidualbottleneck$; and $channel - scaling \in 0.5, 1.0$. The search space options were expanded for 7 blocks of Efficientnet-b0 [25] model, making total search complexity to be $128 * 7 = 896$ blocks. Here, we performed the evolutionary search based on NSGA-2 algorithm for 100 population size and 30 steps. The architecture search for object detection performed by us was completely based on MSCOCO [16] datasets without any imagenet [8] pretraining. In Table 4, we can observe the reduction in computation cost to be around 30% with improvement in the mAP when compared to the mothernet we started with. This proves that DONNAv2 method can be extended to complex vision tasks like object detection using a loss proxy scoring system to obtain the optimized models from an already compressed NAS searched models like EfficientDet-D0. the latency numbers computed for object detection was performed on Samsung galaxy S10 mobile platform, making this a highly efficient hardware friendly model with better performance as compared to the mothernet we started with.

[1] Latency numbers could vary by changing the SNPE SDK version. Here we compute the latency of baseline models with a given SDK version and perform NAS with this particular version to observe the compression in latency.

Table 1: Performances of DONNAv2 compared to other NAS methods

Method	**Granularity**	**Macro-Diversity**	**Search-cost 1 scenario [epochs]**	**Cost/ Scenario 4 scenarios [epochs]**	**Cost/ Scenario ∞ scenarios [epochs]**
DONNA	block-level	variable	4000 + 10 x 50	1500	500
OFA	layer-level	fixed	1200 + 10 x [25 - 75]	550 - 1050	250 - 750
NSGANetV2	layer-level	fixed	1200 + 10 x [25 - 75]	[550 - 1050]	[250 - 750]
DNA	layer-level	fixed	770 + 10 x 450	4700	4500
MNasNet	block-level	variable	40000 + 10 x 450	44500	44500
DONNAv2 (ours)	**block-level**	**variable**	**1200**	-	**500**

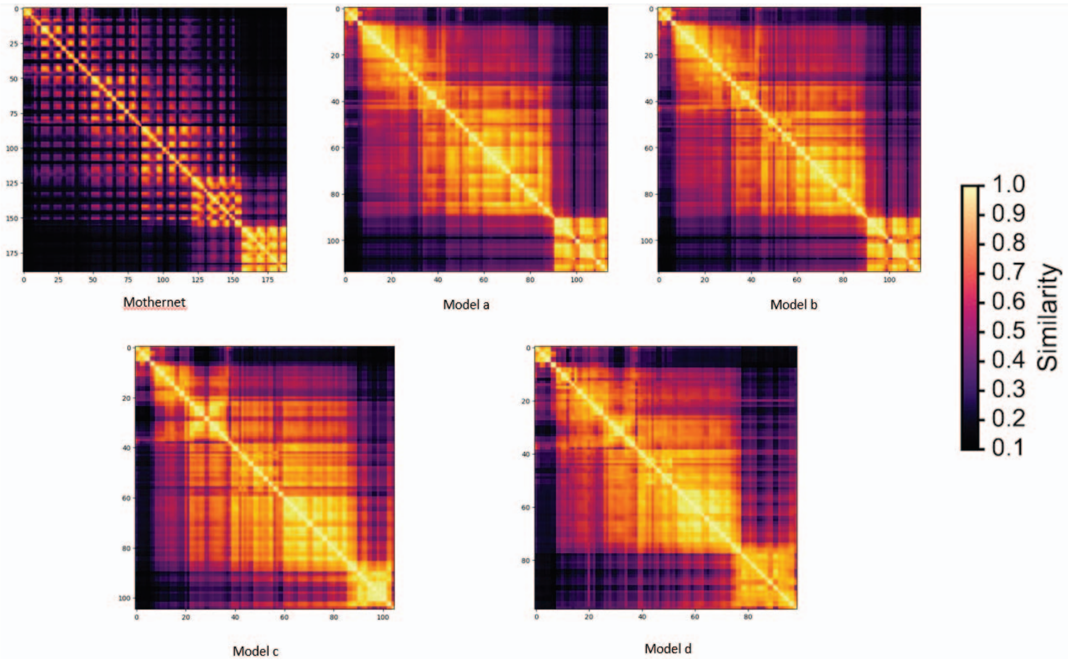


Figure 4: Here, the Mothernet has checkerboard heat map displaying local similarity and model learns different representations across layers. The compressed models (Model a, Model b, Model c and Model d) demonstrate progressively increasing similarity across multiple layers. This suggests that Model a is the best-compressed model in terms of learning distinct representations across layers. This correlates with the performance of the trained model as well as the surrogate loss used in this work.

Table 2: Performances of DONNAv2 optimized models on Image Classification

Model	**Accuracy**	**Latency(in ms)**	**Loss surrogate**
Model a	78.43	1.6	0.171
Model b	77.8	1.47	0.188
Model c	76.36	1.21	0.221
Model d	74.26	1.08	0.267

Table 3: Performances of DONNAv2 optimized models on Image classification

Model	**Accuracy**	**Latency(in ms)**	**Cost/Scenario**
DONNA	77.8	1.6	2500 + 50 + 1e
DONNAv2 (ours)	**77.8**	**1.47**	**50 + 1e**
EfficientNet-B0	77.5	2.0	NA

4.4. Super resolution

For the super-resolution task, we started with a simpler model, Enhanced Deep Residual Networks EDSR [15] that

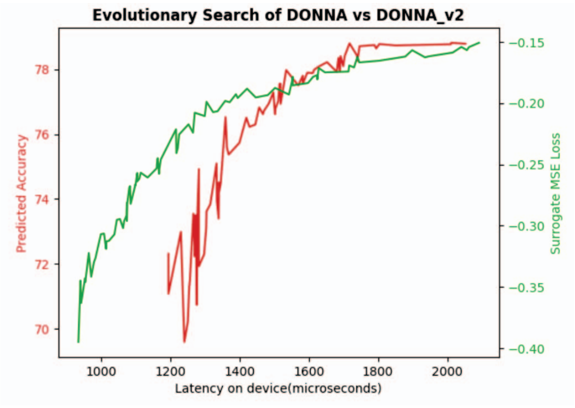


Figure 5: Imagenet Classification Pareto optimal curve of DONNA vs DONNAv2 . The red plot which is the left Y-axis is representative of predicted accuracy vs latency as described in DONNA [20] and the green plot is the surrogate MSE loss vs latency **(our proposed method)**. Both the measurements of latency are performed on Samsung Galaxy S10 mobile platform. DONNAv2 search identifies models across a similar latency spread as in the case of DONNA.

Table 4: Performances of DONNAv2 optimized models on Object Detection

Model	mAP	Latency(in ms)	Cost/Scenario
DONNA	35.1	1.98	2500 + 310 + 1e
DONNAv2 (ours)	**34.8**	**1.98**	**310 + 1e**
EfficientDet-D0	33.4	2.792	NA

has a ResNet-like [12] backbone for image super-resolution task. The search space for this model comprised of searching for two blocks based on Resnet Bottleneck style architecture and one head. The search space options for the Resnet style blocks comprised of $depth \in 1, 2, 3, 4, 6, 8$ along with input channels and bottleneck channels. The search space options for head were different comprising of kernel sizes and upscaling options. This experiment highlights the use-case that proves that the blocks we chose to optimize need not be similar in architecture to be searched over. We support varying macro-architectural parameters such as layer types, activations and attention mechanisms, as well as micro-architectural parameters such as block repeats, kernel sizes and expansion rates. Efficient model search for EDSR using DONNA and DONNAv2 resulted in models with comparable performance. However, with DONNAv2 arrived at the efficient EDSR model with 30% reduction in computational cost. REDS [21] is a small dataset and the finetuning requires 3125 epochs. To estimate an accuracy predictor for Donna, we subsample and finetune 34 candidate models. DONNAv2 avoids fine-tuning models to estimate the performance of candidate models in the search space. Super-resolution is also one of the dense vision tasks with varying block architectures that was able to converge to optimal models using DONNAv2 search algorithm.

Table 5: Performances of DONNAv2 optimized models on Super-Resolution

Model	PSNR(in dB)	Latency(in ms)	Cost/Scenario
DONNA	28.36	8.68	106250 + 3125 + 1e
DONNAv2 (ours)	**28.44**	**7.5**	**3125 + 1e**
EDSR	28.6	16.7	NA

4.5. Image Denoising

For image denoising tasks, one of the most popular architectures is the UNet [22]. To demonstrate the capability of DONNAv2 on image denoising, we chose to optimize a UNet-based multi-stage model NAFNet [6]. NAFnet is one of the state-of-the-art models for image denoising. The evolutionary search over architecture search space with hardware agnostic metrics (Macs count) helped in identifying efficient denoising models with minimal performance degradation. This also demonstrates the efficacy of DONNAv2 for flops-based model search. The optimization strategy could be varied based on use-cases and this is one of the examples proving that DONNAv2 can be performed for a flops-based search strategy as well. Note that NAFNet itself is a lightweight design which added to the difficulty of compressing the model furthermore using NAS. But still, DONNAv2 was able to achieve almost 40% MAC reduction with only about 0.3 PSNR degradation. It is also one of the complexed vision tasks on which very few NAS methodologies have been applied and proved their efficacy against.

Table 6: Performances of DONNAv2 optimized models on Image Denoising

Model	GMACs	PSNR	Cost/Scenario
DONNAv2 (ours)	**40.173**	**39.9895**	540+
Stage-1 NAFNet	63.6	40.3045	NA

4.6. Multi-Task Network: YOLOP

For many vision applications, multi-task networks are being deployed and one such widely used model in the autonomous driving industry is YOLOP [29]. YOLOP has three tasks: traffic light detection, driving area segmentation, and lane segmentation. The architecture of the model consists of an encoder model which forms the backbone of

Table 7: Performances of DONNAv2 optimized models on Multi-task Networks

Model	Detection mAP	Lane segmentation mIOU	Driving segmentation area mIOU	GMACs	Cost/Scenario
DONNAv2 (ours) With backbone alone	74.7	62.5	90.7	12.4	240 + 1e
DONNAv2 (ours) With head included	**75**	**62.8**	**91**	**12.4**	**240 + 1e**
YOLOp	75.6	62.5	91.5	15.5	240

the network and three heads for each of the tasks. The backbone of YOLOP model comprises of five BottleneckCSP blocks, the object detection head comprises of three BottleneckCSP blocks and the segmentation heads comprise of two BottleneckCSP blocks each. This shows that the computational complexity of the model is spread throughout the model. Here, we explored two approaches to find an efficient compressed model. In the first approach, we compressed only the backbone and in the second approach, the NAS search covered both the backbone and head. In both experiments DONNAv2 helps us come up with networks that are 20-35 % compressed without significant performance degradation across tasks. The dataset used for the YOLOP network is the BDD100K [30] dataset. When we compare the compression approaches, as expected, compressing the backbone alone degrades performance across all three tasks when compared with jointly compressing the backbone and the heads. This highlights that when both backbone and heads were searched over, the backbone retained the components needed for accuracy boost and the compression was obtained from the heads as well. This is one of the highly complex model to be searched and it also proves our DONNAv2 can search over segmentation tasks along with multiple head in the tasks as well.

5. Conclusion

In this paper, we have explored the efficacy of the surrogate measure and demonstrated a ten-fold reduction in computational complexity for NAS across widely varying learning tasks. It is of great advantage to researchers to be able to perform NAS searches utilizing very few GPU resources. Furthermore, it is important to note that DONNAv2 came up with efficient models while maintaining accuracy across all learning tasks we have explored. Our DONNAv2 was tested extensively across wide range of complex and dense vision tasks and our experimental studies have shown that DONNAv2 has a significant computational advantage for large ImageNet scale training data. In summary, DONNAv2 provides an efficient NAS approach to building a surrogate performance measure and introduced a novel block filtering approach to improve the quality of models obtained in the evolutionary search step. DONNAv2 has introduced a reliable proxy method that not only makes the NAS faster but can also be applied across wide range of tasks. The limitations of this paper lies in the fact that it is empirically found metric that perform on par or better than the accuracy predictor. In future work, we would like to evaluate the limitations of the metric, if found any.

References

[1] Mohamed S Abdelfattah, Abhinav Mehrotra, Łukasz Dudziak, and Nicholas Donald Lane. Zero-cost proxies for lightweight {nas}. In *International Conference on Learning Representations*, 2021. 2

[2] Cody Blakeney, Xiaomin Li, Yan Yan, and Ziliang Zong. Parallel blockwise knowledge distillation for deep neural network compression, 2020. 1

[3] Han Cai, Chuang Gan, Tianzhe Wang, Zhekai Zhang, and Song Han. Once-for-all: Train one network and specialize it for efficient deployment, 2019. 2

[4] Han Cai, Ligeng Zhu, and Song Han. Proxylessnas: Direct neural architecture search on target task and hardware, 2018. 2

[5] Thomas Chun Pong Chau, Łukasz Dudziak, Hongkai Wen, Nicholas Donald Lane, and Mohamed S Abdelfattah. Blox: Macro neural architecture search benchmark and algorithms, 2022. 2

[6] Liangyu Chen, Xiaojie Chu, Xiangyu Zhang, and Jian Sun. Simple baselines for image restoration, 2022. 7

[7] Kalyanmoy Deb, Amrit Pratap, Sameer Agarwal, and TAMT Meyarivan. A fast and elitist multiobjective genetic algorithm: Nsga-ii. *IEEE transactions on evolutionary computation*, 6(2):182–197, 2002. 5

[8] Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. Imagenet: A large-scale hierarchical image database. In *2009 IEEE Conference on Computer Vision and Pattern Recognition*, pages 248–255, 2009. 4, 5

[9] Łukasz Dudziak, Thomas Chau, Mohamed S. Abdelfattah, Royson Lee, Hyeji Kim, and Nicholas D. Lane. Brp-nas: Prediction-based nas using gcns, 2020. 1

[10] Yicheng Fan, Dana Alon, Jingyue Shen, Daiyi Peng, Keshav Kumar, Yun Long, Xin Wang, Fotis Iliopoulos, Da-Cheng Juan, and Erik Vee. Layernas: Neural architecture search in polynomial complexity, 2023. 1, 2

[11] Ronghao Guo, Chen Lin, Chuming Li, Keyu Tian, Ming Sun, Lu Sheng, and Junjie Yan. Powering one-shot topological nas with stabilized share-parameter proxy, 2020. 2

[12] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition, 2015. 7

[13] Simon Kornblith, Mohammad Norouzi, Honglak Lee, and Geoffrey Hinton. Similarity of neural network representations revisited, 2019. 5

[14] Changlin Li, Jiefeng Peng, Liuchun Yuan, Guangrun Wang, Xiaodan Liang, Liang Lin, and Xiaojun Chang. Blockwisely supervised neural architecture search with knowledge distillation, 2019. 2

[15] Bee Lim, Sanghyun Son, Heewon Kim, Seungjun Nah, and Kyoung Mu Lee. Enhanced deep residual networks for single image super-resolution, 2017. 6

[16] Tsung-Yi Lin, Michael Maire, Serge Belongie, Lubomir Bourdev, Ross Girshick, James Hays, Pietro Perona, Deva Ramanan, C. Lawrence Zitnick, and Piotr Dollár. Microsoft coco: Common objects in context, 2014. 5

[17] Hanxiao Liu, Karen Simonyan, and Yiming Yang. Darts: Differentiable architecture search, 2018. 2

[18] Jieru Mei, Yingwei Li, Xiaochen Lian, Xiaojie Jin, Linjie Yang, Alan Yuille, and Jianchao Yang. Atomnas: Fine-grained end-to-end neural architecture search, 2019. 2

[19] Pavlo Molchanov, Jimmy Hall, Hongxu Yin, Jan Kautz, Nicolo Fusi, and Arash Vahdat. Hardware-aware network transformation, 2022. 2

[20] Bert Moons, Parham Noorzad, Andrii Skliar, Giovanni Mariani, Dushyant Mehta, Chris Lott, and Tijmen Blankevoort. Distilling optimal neural networks: Rapid search in diverse spaces, 2020. 1, 2, 4, 5, 7

[21] Seungjun Nah, Sungyong Baik, Seokil Hong, Gyeongsik Moon, Sanghyun Son, Radu Timofte, and Kyoung Mu Lee. Ntire 2019 challenge on video deblurring and super-resolution: Dataset and study. In *CVPR Workshops*, June 2019. 7

[22] Olaf Ronneberger, Philipp Fischer, and Thomas Brox. U-net: Convolutional networks for biomedical image segmentation, 2015. 7

[23] Dimitrios Stamoulis, Ruizhou Ding, Di Wang, Dimitrios Lymberopoulos, Bodhi Priyantha, Jie Liu, and Diana Marculescu. Single-path nas: Designing hardware-efficient convnets in less than 4 hours, 2019. 2

[24] Mingxing Tan, Bo Chen, Ruoming Pang, Vijay Vasudevan, Mark Sandler, Andrew Howard, and Quoc V. Le. Mnasnet: Platform-aware neural architecture search for mobile, 2018. 2

[25] Mingxing Tan and Quoc V. Le. Efficientnet: Rethinking model scaling for convolutional neural networks, 2020. 4, 5

[26] Mingxing Tan, Ruoming Pang, and Quoc V. Le. Efficientdet: Scalable and efficient object detection, 2020. 5

[27] Alvin Wan, Xiaoliang Dai, Peizhao Zhang, Zijian He, Yuandong Tian, Saining Xie, Bichen Wu, Matthew Yu, Tao Xu, Kan Chen, Peter Vajda, and Joseph E. Gonzalez. Fbnetv2: Differentiable neural architecture search for spatial and channel dimensions, 2020. 2

[28] Bichen Wu, Xiaoliang Dai, Peizhao Zhang, Yanghan Wang, Fei Sun, Yiming Wu, Yuandong Tian, Peter Vajda, Yangqing Jia, and Kurt Keutzer. Fbnet: Hardware-aware efficient convnet design via differentiable neural architecture search, 2018. 2

[29] Dong Wu, Man-Wen Liao, Wei-Tian Zhang, Xing-Gang Wang, Xiang Bai, Wen-Qing Cheng, and Wen-Yu Liu. YOLOP: You only look once for panoptic driving perception. *Machine Intelligence Research*, nov 2022. 7

[30] Fisher Yu, Haofeng Chen, Xin Wang, Wenqi Xian, Yingying Chen, Fangchen Liu, Vashisht Madhavan, and Trevor Darrell. Bdd100k: A diverse driving dataset for heterogeneous multitask learning, 2020. 8

[31] Barret Zoph and Quoc V. Le. Neural architecture search with reinforcement learning, 2016. 2

[32] Barret Zoph, Vijay Vasudevan, Jonathon Shlens, and Quoc V. Le. Learning transferable architectures for scalable image recognition, 2017. 2